Manuel d'utilisation de l'API permettant l'utilisation de services web RECORD

Meradi Rabah

10 juillet 2014

Résumé

Ce manuel d'utilisation présente les fonctionnalités de l'API permettant l'utilisation des services web de la plate-forme RECORD qui ont été développés pour les besoins de la plate-forme MEANS et permettant de lancer une étude de la durabilité environnementale en utilisant le modèle MicMac.

Dans ce document sont présentées uniquement les fonctionnalités de l'API pour savoir comment utiliser les services web RECORD veuillez lire la documentation de ceux-ci.

1 Installation

L'API requiert d'avoir la version 1.7 ou une version antérieure de Java. Pour Utiliser l'API il suffit de télécharger et d'ajouter le fichier jar de l'API à votre projet. Vous pouvez télécharger soit le fichier qui inclut toutes les dépendances ou celui qui contient uniquement le code de cet API. Il faut ajouter à votre projet les libraires suivantes afin de pouvoir utiliser le jar contenant que le code de l'API.

1. jackson-core (version 2.4.0)
2. jackson-databind (version 2.3.3)
3. commons-lang3 (version 3.0)
4. jackson-annotations (version 2.4.1)
5. validation-api (version 1.0)
6. commons-codec(version 1.4)
7. junit (version 4.11) : uniquement pour pouvoir lancer les tests
8. jsonassert (1.2.3) : uniquement pour pouvoir lancer les tests

2 Fonctionnalités

L'API permet de s'authentifier au sein des services web RECORD. Il est possible ensuite de lancer une simulation et récupérer le résultat de simulation. Il faut instancier la classe API en spécifiant l'adresse web des service web ainsi que le port s'il est différent de 80. Vous avez alors ensuite accès à toutes les fonctionnalités expliquées ci-dessous.

```
import record.ws.api;

//Instanciation en spécifiant que l'url
API api = new API("http://147.99.96.186");

//Ou avec le port si différent de 80
API api = new API("http://147.99.96.186", 24000);
```

Exemple 1 – Instanciation de la classe API

2.1 Authentification

L'API fournit la fonction *login* qui permet de s'authentifier au sein des services web RECORD. Il est obligatoire de fournir le nom d'utilisateur et le mot de passe qui vous ont été déjà attribués.
En cas de succès, la fonction récupère le token envoyé par les services web et initialise l'attribut *token* de la classe API. En cas d'erreur, elle lève une exception de type *FailedLoginException*.
La durée de vie du token est gérée du côté des services web. Une fois que cette durée est terminée, on doit rappeler la fonction *login* afin d'obtenir un nouveau *token*. On peut utiliser la fonction *isLogged* de l'API pour vérifier si le *token* est toujours valide ou s'il faut en obtenir un autre.

2.1.1 Exemple

```
//...

API api = new API(url);

try {
   //authentification
   api.login(username, password);
} catch (IOException e) {
// erreur connexion
} catch (FailedLoginException e) {
//erreur d'authentification
}
```

Exemple 2 – Authentification au sein des services web

2.2 Simulation

L'API fournit la fonction *simuler* qui permet de lancer la simulation d'un modèle RECORD et récupérer le résultat de simulation. Il faut d'abord s'authentifier (fonction *login*) avant d'appeler cette fonction.
Pour lancer une simulation on fournit le nom du modèle à simuler (chaîne de caractères) et la requête (objet *Requete*). On construit la requête en donnant les informations nécessaires pour le lancement de la simulation. Actuellement le seul modèle RECORD disponible est le modèle MicMac. L'API dispose alors des deux classes : RequeteMicMac et ResultatMicMac qui sont adaptées pour le modèle MicMac. On peut toujours lancer la simulation d'un nouveau modèle. Pour cela, soit on utilise les objets génériques Requete et Resultat ou on crée des classes qui sont adaptées pour le nouveau modèle. Les classes crées pour le nouveau modèle doivent hériter des classes génériques.
En cas d'erreur une exception est levée. L'erreur peut-être dû à une erreur de connexion, d'authentification ou de simulation.

2.2.1 Exemple

```java
//...

API api = new API(url);
//On construit notre requête
Requete requete = new Requete("BleDur", "Tournesol", "Moutarde",
              "Auzeville");
//L'objet où sera stocké le résultat
ResultatMicMac res = null;
try {
    //On s'authentifie d'abord
    api.login(username, password);
    //On lance la simulation de modèle (ici c'est le modèle MicMac)
    res = api.simuler("MicMac", requete, ResultatMicMac.class);
    //On affiche le résultat
    System.out.println(res);
    //On affiche le résultat sous le format json
    printAsJson(res);
} catch (IOException e) {
    //Erreur de connexion
} catch (NonAuthentifie e) {
    //Erreur d'authentification
} catch (FailedLoginException e) {
    //Erreur d'authentification
} catch (ErreurSimulation e) {
    //Erreur de simulation
}
```

Exemple 3 – Lancement de la simulation de modèle MicMac

3 Résultat

La classe *ResultatMicMac* contient six attributs qui regroupent toutes les informations retournées par les services web. Les six attributs sont :

fluxPolluant
: Il s'agit d'une liste d'objet FluxPolluant Cet attribut contient les valeurs des flux polluants de la plate-forme MEANS. Certains flux polluants de la plate-forme MEANS ne sont pas disponibles et seront calculés par MEANS en utilisant des données des autres variables (cf.

bilan de la réunion de 23 mai 2014).

culturePrecedente
Il est de type Culture. Il contient les informations de la culture précédente.

cultureIntermediaire
Il est de type Culture. Il contient les informations sur la culture intermédiaire.

culturePrincipale
Il est de type Culture. Il contient des information sur la culture principale.

coProduit
Il est de type CoProduit. Il contient les informations du co-produit.

solErosion
Il est de type SolErosion. Il contient les informations sur la variable sol érosion.

Ces six attributs sont expliqués dans les sous sections suivantes.

3.1 Flux Polluant

Les données du tableau suivant sont stockées dans l'objet *fluxPolluant* de la classe *ResultatMicMac*. La troisième colonne du tableau explique comment récupérer la valeur d'un flux. On utilise la fonction *getFluxValeurs* de la classe API.

Variable MEANS	Définition	Équivalent dans ***ResultatMicMac***
CO2	Émission de CO2 liée à l'épandage de chaux.	On peut récupérer sa valeur via la fonction *getFluxValeurs()* avec le nom *CO2Chaux*.
CO2	Émission de CO2 liée à l'épandage d'urée.	On peut récupérer sa valeur via la fonction *getFluxValeurs()* avec le nom *CO2Uree*.
QCO2sol	*Cumulative amount of c-co2 derived from soil heterotrophic respiration*	On peut récupérer sa valeur via la fonction *getFluxValeurs()* avec le nom *QCO2sol*.

NH3 (ammoniac)	Volatilisation d'azote suite à un épandage d'engrais.	On peut récupérer sa valeur via la fonction *getFluxValeurs()* avec le nom *NH3*.
NO3 (nitrates)	Lixiviation de l'azote. Agrégation des quantités journalières sur une durée différente de celle de la simulation. Durée de simulation : de la date du semis de la culture étudiée à la date de semis de la culture suivante.	On peut récupérer sa valeur via la fonction *getFluxValeurs()* avec le nom *NO3 (QLES)*.
N2O direct (Protoxyde d'azote, direct)	Volatilisation d'azote suite à un épandage d'engrais.	On peut récupérer sa valeur via la fonction *getFluxValeurs()* avec le nom *N2ODirect*.
N2O indirect (Protoxyde d'azote, indirect)	Transformation d'ammoniac et des nitrates émis pendant la culture en N2O	On peut récupérer sa valeur via la fonction *getFluxValeurs()* avec le nom *N2OIndirect*.
NO	Émission de monoxyde d'azote suite à un épandage d'engrais.	Cette variable n'est pas calculée par le service web. Pour permettre de la calculer on fournit les apports d'engrais pour la culture principale. Pour les apports d'engrais voir section **??**, activité de fertilisation.

Phosphate lixivié	Lixiviation de phosphate.	Le service web ne calcule pas cette variable. Pour permettre de la calculer on fournit les apports de phosphate. Voir section **??**, activité Traitements phytosanitaires.
Phosphate entraîné par ruissellement	Phosphate dissous dans les eaux de ruissellement.	Même remarque que pour le Phosphate lixivié.
Phosphore érodé	Phosphore entraîné sous forme particulaire.	Même remarque que pour le Phosphate lixivié.
Lixiviation de 7 éléments traces métalliques (Cd, Cu, Cr, Ni, Hg, Pb, Zn)	Quantité sur la durée de simulation : de la récolte de la culture précédente à la récolte de la culture étudiée attention il s'agit de la masse de phosphate et non de la masse de phosphore.	Cette variable n'est pas calculée par le service web. On utilise la date de récolte de la culture principale et de la culture précédente. Pour ces deux informations voir section **??**, **??**, activité Récolte.
Émissions de 7 éléments traces métalliques par érosion (Cd, Cu, Cr, Ni, Hg, Pb, Zn)	Quantité sur la durée de simulation : de la récolte de la culture précédente à la récolte de la culture étudiée attention il s'agit de la masse de phosphate et non de la masse de phosphore.	Même remarque que pour la variable précédente.

Accumulation de 7 éléments traces métalliques (Cd, Cu, Cr, Ni, Hg, Pb, Zn) dans les sols.	Quantité sur la durée de simulation : de la récolte de la culture précédente à la récolte de la culture étudiée attention il s'agit de la masse de phosphate et non de la masse de phosphore.	Cette variable n'est pas calculée directement par les services web. Les informations nécessaires pour son calcul sont : la quantité de semence de la culture principale, la quantité d'engrais utilisée, la quantité des pesticides utilisée, le rendement de la culture principale et le rendement de co-produit. Pour ces données voir les sections **??**, **??**.

Table 1 – Récupération des valeurs des flux polluants

3.1.1 Exemple

```
//...

//On récupère le résultat de simulation de MicMac
ResultatMicMac resultat = api.simuler("MicMac", requete,
  ResultatMicMac.class);
//On récupère les valeurs des émission de CO2 liée à l'épandage
  de chaux
List<Valeur> co2ChauxValeurs = resultat.
                              getFluxValeurs("CO2Chaux");
//On affiche les valeurs
```

Exemple 4 – Récupération des valeurs d'un flux polluant

3.2 Caractérisation du système de production

3.2.1 Activités

Il existe dans l'API une classe *Activites* qui contient l'ensemble des activités agronomiques prises en charge par le modèle MicMac. Il existe alors une classe pour chaque activité et la classe contient les informations de cette activité.

Les différentes activités sont expliquées dans le tableau ??. Ces classes sont regroupées dans le *package record.ws.api.resultat.activites*. La classe Culture contient un attribut de type Activites.

Variable MEANS	**Équivalent dans l'API**
Activité de Semis (Classe Semis)	
Date de semis	Champ *date*.
Quantité de semence	Champ *quantite*.
Opération agricole	Champ *opAgricole*.
Nombre de passages	Champ *nbPassages*.
Activité de Fertilisation (classe Fertilisation)	
Date	Champ *date*.
Nom	Champ *typeN*.
Quantité de fertilisant	Champ *quantite*.
Opération agricole	Champ *opAgricole*.
Nombre de passages	Champ *nbPassages*.
Traitements phytosanitaires (classe TraitementsPhytosanitaire)	
Nature	Champ *nom*.
Quantité	Champ *quantite*.
Pourcentage de surface traitée	Champ *pourcentage*.
Opération agricole	Champ *opAgricole*.
Nombre de passages	Champ *nbPassages*.
Irrigation/Fertirrigation (classe IrrigationFertirrigation)	
Date	Champ *date*.
Source d'énergie	Champ *sourceEnergie*.
Quantité d'eau	Champ *quantiteEau*.
Matériel	Champ *materiel*.
Travail du sol (classe TravailSol)	
Date	Champ *date*.
Type	Champ *nom*.
Profondeur	Champ *profondeur*.
Nombre de passages	Champ *nbPassages*.
Récolte (classe Recolte)	

Date de récolte	Champ *date* (type Date).
Opération agricole	Champ *opAgricole* (type OpAgricole).
Opération post-récolte (classe PostRecolte)	
Nom	Champ *nom* (de type String).
Quantité	Champ *quantite* (de type long)

TABLE 2 – Les différents activités disponibles dans l'API

3.2.2 Culture précédente

Les données du tableau suivant sont stockées dans l'attribut *culturePrecedente* de la classe *ResultatMicMac*. L'attribut culturePrecedente est de type Culture. Cette contient les activités effectuées sur la culture précédente. Actuellement la seule activité retournée pour la culture précédente est l'activité de semis.

Variable	**Équivalent dans l'API**
Activité de semis	On peut récupérer l'activité de semis via la variable *activites*.

TABLE 3 – Informations concernant la culture précédente

3.2.3 Exemple

```
//...

API api = new API(url);

ResultatMicMac resultat = api.simule("MicMac", requete,
  ResultatMicMac.class);

//On récupère les informations sur la culture précédente
CulturePrecedente culturePrecedente =
  resultat.getCulturePrecedente();

//on récupère l'activité de semis
Semis semis = culturePrecedente.getActivites().getSemis();

//On affiche la date de semis
```

Exemple 5 – Récupération des informations de la culture précédente

3.2.4 Culture intermédiaire

Les données du tableau suivant sont stockées dans l'attribut *cultureIntermediaire* de la classe *ResultatMicMac*. L'attribut cultureIntermediaire est de type Culture. Cette classe contient les activités effectuées sur la culture précédente. Actuellement la seule activité retournée pour la culture intermédiaire est l'activité de récolte.

Donnée	**Équivalent dans l'API**
Le rendement	Champ *rendement*
Activité de récolte	On peut récupérer l'activité de récolte via la variable *activites*.

Table 4 – Informations concernant la culture intermédiaire

3.2.5 Exemple

```java
//...

API api = new API(url);

ResultatMicMac resultat = api.simule("MicMac", requete,
  ResultatMicMac.class);

//On récupère les informations sur la culture précédente
CultureIntermediaire cultureIntermediaire =
  resultat.getCultureIntermediaire();

//on affiche le rendement
System.out.println(cultureIntermediaire.getRendement());

//on récupère l'activité de récolte
Recolte recolte =
  cultureIntermediaire.getActivites().getRecolte();

//On affiche la date de récolte
```

Exemple 6 – Récupération des informations de la culture intermédiaire

3.2.6 Culture principale

Les informations sur la culture principale sont stockées dans la attribut *culturePrincipale* de la classe *ResultatMicMac*. L'attribut est de type Culture et contient le rendement de la culture et la liste de toutes les activités effectuées pour cette culture (Objet Activites). Pour les informations concernant les activités voir la section **??**.

Donnée	Équivalent
Rendement	Champ *rendement*
Activité de Semis	Disponible dans la variable *activites*.
Activité de Fertilisation	Même remarque que précédent.
Traitements phytosanitaires	Même remarque que précédent.

Activité d'Irrigation/ Fertirrigation	Même remarque que précédent.
Travail du sol	Même remarque que précédent.
Activité de Récolte	Même remarque que précédent.
Activité de post-récolte	Même remarque que précédent.

Table 5 – Informations concernant la culture principale

3.2.7 Exemple

```
//...

API api = new API(url);

ResultatMicMac resultat = api.simule("MicMac", requete,
  ResultatMicMac.class);

//On récupère les informations sur la culture précédente
CulturePrincipale culturePrincipale =
  resultat.getCulturePrincipale();

//on affiche le rendement
System.out.println(culturePrincipale.getRendement());

//on récupère l'activité de récolte
Recolte recolte = culturePrincipale.getActivites().getRecolte();

//On affiche la date de récolte
```

Exemple 7 – Récupération des informations de la culture principale

3.2.8 Co-produit

Les données expliquées dans le tableau suivant se trouvent dans la variable *coProduit* de la classe *ResultatMicMac*.

Donnée	Équivalent
Le nom de co-produit	Correspond au champ *nom*.

Son rendement	Correspond au champ *rendement*.
Son contenu en carbone	Correspond au champ *contenuCarbon*.

TABLE 6 – Informations concernant le co-produit

3.2.9 Exemple

```
//...

API api = new API(url);

ResultatMicMac resultat = api.simule("MicMac", requete,
  ResultatMicMac.class);

//On récupère les informations sur la culture précédente
CoProduit coProduit = resultat.getCoProduit();

//On affiche son nom
System.out.println(coProduit.getNom());

//on affiche le rendement
System.out.println(coProduit.getRendement());

//On affiche son contenu en carbone
```

Exemple 8 – Récupération des informations du co-produit

3.2.10 Sol et érosion

Donnée	**Équivalent dans l'API**
Méthode de travail du sol la plus érosive	Champ *methodeTravailSolPlusErosive* (de type String)

TABLE 7 – Informations concernant la variable sol et érosion

3.2.11 Exemple

```
//...

API api = new API(url);

ResultatMicMac resultat = api.simule("MicMac", requete,
  ResultatMicMac.class);

//On récupère les informations sur le sol et érosion
SolErosion solErosion = resultat.getSolErosion();

//On affiche le nom de la méthode de travail du sol la plus
  érosive
```

Exemple 9 – Récupération des informations sur la variable sol et érosion